Some packages relative to machine learning in R

林松江

2005/9/30
Machine learning methods

- **Supervised Learning**, where we get a set of training inputs and outputs
  - classification, regression
    - Tree, SVM, KNN, LDA, ...

- **Unsupervised Learning**, where we are interested in capturing inherent organization in the data
  - clustering, density estimation
    - K-mean, EM, SOM, …
Install new packages “tree”

For example: > install.packages("tree")
Load packages "tree"

- For example: > library(tree)
Help for new packages
Usage for tree: input argument

> help("tree")

R Help on 'tree'

Fit a Classification or Regression Tree

Description:

A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

Usage:

```
tree(formula, data, weights, subset,
    na.action = na.pass, control = tree.control(nobs, ...),
    method = "recursive.partition",
    split = c("deviance", "gini"),
    model = FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)
```

Arguments:

formula: A formula expression. The left-hand-side (response) should be either a numerical vector when a regression tree will be fitted or a factor, when a classification tree is produced.
Usage for tree: Examples


See Also:

'tree.control', 'prune.tree', 'predict.tree', 'snip.tree'

Examples:

```r
library(MASS)
data(cpus)
cpus.ltr <- tree(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus)
cpus.ltr
summary(cpus.ltr)
plot(cpus.ltr); text(cpus.ltr)

data(iris)
ir.tr <- tree(Species ~ ., iris)
ir.tr
summary(ir.tr)
```
Dataset in R

```r
> data()
```

Data sets in package 'datasets':

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirPassengers</td>
<td>Monthly Airline Passenger Numbers 1949-1960</td>
</tr>
<tr>
<td>BJsales</td>
<td>Sales Data with Leading Indicator</td>
</tr>
<tr>
<td>BJsales_lead (BJsales)</td>
<td>Sales Data with Leading Indicator</td>
</tr>
<tr>
<td>BOD</td>
<td>Biochemical Oxygen Demand</td>
</tr>
<tr>
<td>CO2</td>
<td>Carbon Dioxide uptake in grass plants</td>
</tr>
<tr>
<td>ChickWeight</td>
<td>Weight versus age of chicks on different diets</td>
</tr>
<tr>
<td>DNase</td>
<td>Elisa assay of DNase</td>
</tr>
<tr>
<td>EuStockMarkets</td>
<td>Daily Closing Prices of Major European Stock</td>
</tr>
<tr>
<td>Formaldehyde</td>
<td>Determination of Formaldehyde</td>
</tr>
<tr>
<td>HairEyeColor</td>
<td>Hair and Eye Color of Statistics Students</td>
</tr>
<tr>
<td>Harman23.cor</td>
<td>Harman Example 2.3</td>
</tr>
<tr>
<td>Harman74.cor</td>
<td>Harman Example 7.4</td>
</tr>
<tr>
<td>Indometh</td>
<td>Pharmacokinetics of Indomethin</td>
</tr>
<tr>
<td>InsectSprays</td>
<td>Effectiveness of Insect Sprays</td>
</tr>
<tr>
<td>JohnsonJohnson</td>
<td>Quarterly Earnings per Johnson &amp; Johnson Share</td>
</tr>
<tr>
<td>LakeHuron</td>
<td>Level of Lake Huron 1875-1972</td>
</tr>
<tr>
<td>LifeCycleSavings</td>
<td>Intercountry Life-Cycle Savings Data</td>
</tr>
<tr>
<td>Loblolly</td>
<td>Growth of Loblolly pine trees</td>
</tr>
<tr>
<td>Nile</td>
<td>Flow of the River Nile</td>
</tr>
<tr>
<td>Orange</td>
<td>Growth of orange trees</td>
</tr>
<tr>
<td>OrchardSprays</td>
<td>Potency of Orchard Sprays</td>
</tr>
</tbody>
</table>
For example: load "iris" dataset

- > data(iris)
- > ls()
  [1] "iris"
- > iris[1:5,]

<table>
<thead>
<tr>
<th></th>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
</tbody>
</table>
Fit a Classification Tree

> ir.tr <- tree(Species ~ ., iris)
> ir.tr

node, split, n, deviance, yval, (yprob)
  * denotes terminal node

1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
   2) Petal.Length < 2.45 50 0.000 setosa ( 1.00000 0.00000 0.00000 ) *
   3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
     6) Petal.Width < 1.75 54 33.320 versicolor ( 0.00000 0.90741 0.09259 )
       12) Petal.Length < 4.95 48 9.721 versicolor ( 0.00000 0.97917 0.02083 )
       24) Sepal.Length < 5.15 5 5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
       25) Sepal.Length > 5.15 43 0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
     13) Petal.Length > 4.95 6 7.638 virginica ( 0.00000 0.33333 0.66667 ) *
    7) Petal.Width > 1.75 46 9.635 virginica ( 0.00000 0.02174 0.97826 )
   14) Petal.Length < 4.95 6 5.407 virginica ( 0.00000 0.16667 0.83333 ) *
   15) Petal.Length > 4.95 40 0.000 virginica ( 0.00000 0.00000 1.00000 ) *

> summary(ir.tr)

Classification tree:
tree(formula = Species ~ ., data = iris)
Variables actually used in tree construction:
Plot the fitted tree model

- > plot(ir.tr)
- > text(ir.tr)
Predictions from a Fitted Tree Object

> predict(ir.tr, iris[,1:4], type="class")

[1] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[7] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[13] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[19] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[25] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[31] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[37] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[43] setosa  setosa  setosa  setosa  setosa  setosa  setosa
[49] setosa  setosa  versicolor versicolor versicolor versicolor
[55] versicolor versicolor versicolor versicolor versicolor versicolor
[61] versicolor versicolor versicolor versicolor versicolor versicolor
[67] versicolor versicolor versicolor versicolor versicolor versicolor
[73] versicolor versicolor versicolor versicolor versicolor versicolor
[79] versicolor versicolor versicolor versicolor versicolor versicolor
[85] versicolor versicolor versicolor versicolor versicolor versicolor
[91] versicolor versicolor versicolor versicolor versicolor versicolor
[97] versicolor versicolor versicolor versicolor versicolor versicolor
[103] virginica virginica virginica virginica versicolor virginica
[109] virginica virginica virginica virginica virginica virginica
[115] virginica virginica virginica virginica virginica virginica
[121] virginica virginica virginica virginica virginica virginica
[127] virginica virginica virginica virginica virginica virginica
[133] virginica virginica virginica virginica virginica virginica
[139] virginica virginica virginica virginica virginica virginica
[145] virginica virginica virginica virginica virginica virginica
Levels: setosa versicolor virginica
Computing the accuracy

- \( \text{predict(ir.tr, iris[,1:4], type="class"}) \rightarrow \text{ir.pred} \)
- \( \text{sum(ir.pred==iris[,5])/nrow(iris)} \)

\begin{verbatim}
[1] 0.9733333 accuracy
\end{verbatim}
Package : “e1071”

- Install new packages for
  - > install.packages("e1071")

- Include “svm”, “naiveBayes”, … functions

- Can tunes hyperparameters of statistical methods : “tune” function
Using **svm** in “**e1071**” (1)

```r
> library(e1071); help("svm")
```

**Usage**

```r
## S3 method for class 'formula':
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = 1 / ncol(as.matrix(x)), coef0 =
class.weights = NULL, cache.size = 40, tolerance = 0.001, epsi
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TR
..., subset, na.action = na.omit)
```

**Arguments**

- `formula` a symbolic description of the model to be fit.
- `data` an optional data frame containing the variables in the model. By default the variables are taken from the environment which ‘svm’ is called from.
- `x` a data matrix, a vector, or a sparse matrix (object of class
Using \textbf{svm} in "\texttt{e1071}" (2)

**Fitted a svm model for iris dataset**

\begin{verbatim}
> model <- svm(factor(Species) ~ ., data = iris)
> summary(model)

Call:
  svm(formula = factor(Species) ~ ., data = iris)

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel:  radial
      cost:  1
     gamma:  0.25

Number of Support Vectors:  51

( 8 22 21 )

Number of Classes:  3

Levels:
    setosa versicolor virginica
\end{verbatim}
Using **svm** in "e1071" (3)

- Predict from svm model
  ```
  > x<-iris[,,-5]
  > y<-iris[,5]
  > predict(model,x)->pred
  > print(pred)
  ```
- Compute accuracy for prediction
  ```
  > sum(pred==y)/length(y)
  [1] 0.9733333
  ```
Using **svm** in "e1071" (4)

**Using Linear kernel to fit svm model**

```r
> model <- svm(factor(Species) ~ ., data = iris, kernel="linear")
> summary(model)

Call:
svm(formula = factor(Species) ~ ., data = iris, kernel = "linear")

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel:  linear
  cost: 1
  gamma: 0.25

Number of Support Vectors:  29

  ( 2 15 12 )

Number of Classes:  3

Levels:
setosa versicolor virginica
Using \textbf{svm} in “e1071” (5)

- Predict from svm \textit{Linear} model
  \begin{verbatim}
  > predict(model,x)->linear.pred
  > sum(linear.pred==y)/length(y)
  [1] 0.9666667
  \end{verbatim}

- Confusion matrix
  \begin{verbatim}
  > table(linear.pred, y)
  linear.pred  setosa versicolor virginica
    setosa        50         0         0
    versicolor      0        46         1
    virginica       0         4        49
  \end{verbatim}
Parameter Tuning

Function: "tune" in "e1071" (1/2)

Usage:

```r
  tune(method, train.x, train.y = NULL, data = list(), validation.x =
       NULL, validation.y = NULL, ranges = NULL, predict.func = predict,
       tunecontrol = tune.control(), ...) 
  best.tune(...) 
```

Arguments:

- `method`: function to be tuned.

- `train.x`: either a formula or a matrix of predictors.

- `train.y`: the response variable if `train.x` is a predictor matrix.
  Ignored if `train.x` is a formula.

- `data`: data, if a formula interface is used. Ignored, if predictor
  matrix and response are supplied directly.

- `validation.x`: an optional validation set. Depending on whether a
  formula interface is used or not, the response can be
  included in `validation.x` or separately specified using
  `validation.y`.

- `validation.y`: if no formula interface is used, the response of the
  (optional) validation set.

- `ranges`: a named list of parameter vectors spanning the sampling
  space. The vectors will usually be created by `seq`. 
Parameter Tuning

Function: “tune” in “e1071” (2/2)

- tune “svm” for classification with RBF-kernel (default in svm)
  - gamma=0.5,1,2 and cost=4,8,16

```r
> obj <- tune(svm, Species~., data = iris,
+ ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)))
> obj

Parameter tuning of `svm`:

- sampling method: 10-fold cross validation

- best parameters:
  gamma  cost
  0.5     4

- best performance: 0.04
k-Nearest Neighbour Classification: \texttt{knn} in “class” (1/2)

Usage:

\texttt{knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)}

Arguments:

\texttt{train}: matrix or data frame of training set cases.

\texttt{test}: matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.

\texttt{cl}: factor of true classifications of training set

\texttt{k}: number of neighbours considered.

\texttt{l}: minimum vote for definite decision, otherwise 'doubt'. (More precisely, less than 'k-1' dissenting votes are allowed, even if 'k' is increased by ties.)

\texttt{prob}: If this is true, the proportion of the votes for the winning class are returned as attribute 'prob'.

\texttt{use.all}: controls handling of ties. If true, all distances equal to the 'k'\textquotesingle{}th largest are included. If false, a random selection of distances equal to the 'k'\textquotesingle{}th is chosen to use exactly 'k' neighbours.
k-Nearest Neighbour Classification: \textbf{knn} in "class" (2/2)

- For example: iris dataset
  
  \begin{verbatim}
  > trn.x<-iris[,,-5] ; tst.x<-iris[,,-5]
  > trn.y<-iris[,5] ; tst.y<-iris[,5]
  \end{verbatim}

- Fit the knn with k=3
  
  \begin{verbatim}
  > knn(trn.x, tst.x, trn.y, k = 3, prob=TRUE)->knn.pred
  > print(knn.pred)
  \end{verbatim}

- Confusion matrix
  
  \begin{verbatim}
  > table(knn.pred, tst.y)
  knn.pred   setosa versicolor virginica
  setosa    50      0         0
  versicolor 0      47        3
  virginica 0       3       47
  \end{verbatim}
Evaluation: Receiver Operating Characteristic (ROC) curve analysis

- Plot ROC Curve: "ROCR" package
  ```
  > library(ROCR); data(ROCR.simple)
  > pred<-prediction( ROCR.simple$predictions, ROCR.simple$labels)
  > perf <- performance(pred,"tpr","fpr")
  > plot(perf)
  ```
Packages for Machine learning

- For classification
  - tree in “tree”
  - svm in “e1071”
  - knn in “class”
  - lda in “MASS”
  - adaboost in “boost”

- For clustering
  - kmean in “stats”

- Other useful packages
  - caTools, kernlab, mlbench, cluster, …